2019 Honors App Development Benchmark #1

In this benchmark, I have set up the interface of an application—your task is to implement the behavior. The app is structured like a mini "treasure hunt" where users must navigate through three "puzzle" screens. At the final screen, they will have either successfully or unsuccessfully revealed a treasure, depending on their puzzle results.

Each screen is described below. You will not have to change anything that is provided within the block code for you, but instead will add code to implement functionality.

Screen1

The function of this screen is just to introduce the game. You have only one task here:

- ☐ When Button1 is clicked, the app should open the "Puzzle1" screen.

Puzzle1

This screen more or less randomly generates a math problem, requiring the user to get the correct answer to earn a point for the screen. You must:

- ☐ Randomly select an item from the list of available operations.
- ☐ Randomly generate two numbers, 0 through 9, to be the operands for the selected operation.
- ☐ Modify the problemLabel text to look like the problem at hand. For example, if the numbers 2 and 7 are generated, and multiplication randomly selected, the label should read "2x7=?".
- ☐ Ensure that the keyboard that pops up for the TextBox is the numeric keyboard (you don't have to perform additional checks on the input type beyond this)
- ☐ When the nextButton is clicked, the program should check if the entered text is the correct solution to the problem, and go to the "Puzzle2" screen.
    - o Please note, you will have to somehow indicate here to "Puzzle2" whether or not the point was earned for "Puzzle1".

Puzzle2

This screen displays a set of numbers, and the user must correctly select the seventh in a ListPicker. You must:

- ☐ Ensure the first possible selection for the ListPicker is the phrase "Which was the seventh number displayed?". I WON'T change the order of the numbers in the label.
- ☐ In some way, load the values 0 through 9 (in that order into the ListPicker so they may be selected from when it is clicked. You will have to do this "efficiently" to receive full credit.

- Make sure the numbers are visible when the Screen is opened, but as soon as the ListPicker is clicked on, they should be hidden for the remainder of the user's time on this screen.

- After an element is selected from the ListPicker, the program should award a point for the screen if the selection was the same as the seventh number in the label (i.e., the number 9). The program should then move on to the "Puzzle3" screen.

  - Please note, you will have to somehow indicate here to "Puzzle3" whether or not the point was earned for "Puzzle2" (and "Puzzle1").

Puzzle3

This screen is not intelligence based—it's all luck for the user. They must click one of nine buttons, and have a 1 in 9 chance of it being the "correct" one selected by the computer. You must:

- Have the program randomly select one of the 9 numbered buttons to be the "winning" one.

- When the user clicks one of the 9 numbered buttons, a few things must happen:

  - If the clicked button is the "correct" randomly selected one, it should turn green.

  - If the clicked button is NOT the "correct" randomly selected one, it should turn red.

  - Regardless of correctness of the click, disable all numbered buttons when one is clicked.

- When the user clicks the nextButton, move on to "TreasureScreen".

  - Please note, you will have to somehow indicate here to "TreasureScreen" whether or not the point was earned for "Puzzle3" (and all previous screens).

TreasureScreen

This is the results screen. You must:

- If all three puzzles were successfully solved, the resultImage should be set to "treasurechest.png", and the resultLabel to some congratulatory label.

- If all three puzzles were NOT successfully solved, the resultImage should be set to "lockedtreasurechest.png", and the resultLabel to something indicating the user lost.

- When the restartButton is clicked, the app should return to Screen1.

  - Please note: old runs of the game should NOT affect new ones. It should be a true restart if this button is clicked.

When you have implemented all required behavior, you may use the rest of class to test your app. Submit your final .aia file on Teams, renamed to TreasureHunt*.aia, with the * replaced by your last name.